

Network Construction with Ordered Constraints

Yi Huang, Mano Vikash Janardhanan, and Lev Reyzin*

University of Illinois at Chicago,
 {yhuang89, mjanar2, lreyzin}@uic.edu

Abstract. In this paper, we study the problem of constructing a network by observing ordered connectivity constraints, which we define herein. These ordered constraints are made to capture realistic properties of real-world problems that are not reflected in previous, more general models. We give hardness of approximation results and nearly-matching upper bounds for the offline problem, and we study the online problem in both general graphs and restricted sub-classes. In the online problem, for general graphs, we give exponentially better upper bounds than exist for algorithms for general connectivity problems. For the restricted classes of stars and paths we are able to find algorithms with optimal competitive ratios, the latter of which involve analysis using a potential function defined over pq-trees.

Keywords: graph connectivity, network construction, ordered connectivity constraints, pq-trees

1 Introduction

In this paper, we study the problem of recovering a network after observing how information propagates through the network. Consider how a tweet (through “retweeting” or via other means) propagates through the Twitter network – we can observe the identities of the people who have retweeted it and the timestamps when they did so, but may not know, for a fixed user, via whom he got the original tweet. So we see a chain of users for a given tweet. This chain is semi-ordered in the sense that, each user retweets from some one before him in the chain, but not necessarily the one directly before him. Similarly, when a virus such as Ebola spreads, each new patient in an outbreak is infected from someone who has previously been infected, but it is often not immediately clear from whom.

In a graphical social network model with nodes representing users and edges representing links, an “outbreak” illustrated above is captured exactly by the concept of an **ordered constraint** which we will define formally below. One could hope to be able to learn something about the structure of the network by observing repeated outbreaks, or a sequence of ordered constraints.

Formally we call our problem **Network Construction with Ordered Constraints** and define it as follows. Let $V = \{v_1, \dots, v_n\}$ be a set of vertices. An **ordered constraint** \mathcal{O} is an ordering on a subset of V of size $s \geq 2$.

* Supported in part by ARO grant 66497-NS.

The constraint $\mathcal{O} = (v_{k_1}, \dots, v_{k_s})$ is satisfied if for any $2 \leq i \leq s$, there exists at least one $1 \leq j < i$ such that the edge $e = \{v_{k_j}, v_{k_i}\}$ is included in a solution. Given a collection of ordered constraints $\{\mathcal{O}_1, \dots, \mathcal{O}_r\}$, the task is to construct a set E of edges among the vertices V such that all the ordered constraints are satisfied and $|E|$ is minimized.

We can see that our newly defined problem resides in a middle ground between path constraints, which are too rigid to be very interesting, and the well-studied subgraph connectivity constraints [7,17,18], which are more relaxed. The established subgraph connectivity constraints problem involves getting an arbitrary collection of connectivity constraints $\{S_1, \dots, S_r\}$ where each $S_i \subset V$ and requires vertices in a given constraint to form a connected induced subgraph. The task is to construct a set E of edges satisfying the connectivity constraints such that $|E|$ is minimized.

We want to point out one key observation relating the ordered constraint to the connectivity constraint – an ordered constraint $\mathcal{O} = (v_{k_1}, \dots, v_{k_s})$ is equivalent to $s-1$ connectivity constraints S_2, \dots, S_s , where $S_i = \{v_{k_1}, \dots, v_{k_i}\}$. We note that this observation plays an important role in several proofs in this paper which employ previous results on subgraph connectivity constraints – in particular, upper bounds from the more general case can be used in the ordered case (with some overhead), and our lower bounds apply to the general problem.

In the offline version of the Network Construction with Ordered Constraints problem, the algorithm is given all of the constraints all at once; in the **online** version of the problem, the constraints are given one by one to the algorithm, and edges must be added to satisfy each new constraint when it is given. Edges cannot be removed.

An algorithm is said to be **c-competitive** if the cost of its solution is less than c times OPT, where OPT is the best solution in hindsight (c is also called the competitive ratio). When we restrict the underlying graph in a problem to be a class of graphs, e.g. trees, we mean all the constraints can be satisfied, in an optimal solution (for the online case, in hindsight), by a graph from that class.

1.1 Past Work

In this paper we study the problem of network construction from ordered constraints. This is an extension of the more general model where constraints come unordered.

For the general problem, Korach and Stern [17] had some of the initial results, in particular for the case where the constraints can be optimally satisfied by a tree, they give a polynomial time algorithm that finds the optimal solution. In subsequent work, in [18] Korach and Stern considered this problem for the even more restricted problem where the optimal solution forms a tree, and all of the connectivity constraints must be satisfied by stars.

Then, Angluin et al. [7] studied the general problem, where there is no restriction on structure of the optimal solution, in both the offline and online settings. In the offline case, they gave nearly matching upper and lower bounds on the hardness of approximation for the problem. In the online case, they give

a $O(n^{2/3} \log^{2/3} n)$ -competitive algorithm against oblivious adversaries; we show that this bound can be drastically improved in the ordered version of the problem. They also characterized special classes of graphs, i.e. stars and paths, which we are also able to do herein for the ordered constraint case. Independently of that work, Chockler et al. [12] also nearly characterized the offline general case.

In a different line of work Alon et al. [3] explore a wide range of network optimization problems; one problem they study involves ensuring that a network with fractional edge weights has a flow of 1 over cuts specified by the constraints. Alon et al. [2] also study approximation algorithms for the Online Set Cover problem which have been shown by Angluin et al [7] to have connections with Network Construction problems.

In related areas, Gupta et al. [16] considered a network design problem for pairwise vertex connectivity constraints. Moulin and Laigret [19] studied network connectivity constraints from an economics perspective. Another motivation for studying this problem is to discover social networks from observations. This and similar problems have also been studied in the learning context [5,6,14,22].

Finally, in query learning, the problem of discovering networks from connectivity queries has been much studied [1,4,8,9,15,21]. In active learning of hidden networks, the object of the algorithm is to learn the network exactly. Our model is similar, except the algorithm only has the constraints it is given, and the task is to output the cheapest network consistent with the constraints.

1.2 Our results

In Section 2, we examine the offline problem, and show that the Network Construction problem is NP-Hard to approximate within a factor of $\Omega(\log n)$. A nearly matching upper bound comes from Theorem 2 of [7].

In Section 3, we study online problem. For problems on n nodes, for r constraints, we give an $O((\log r + \log n) \log n)$ competitive algorithm against oblivious adversaries, and an $\Omega(\log n)$ lower bound (Section 3.1).

Then, for the special cases of stars and paths (Sections 3.2 and 3.3), we find asymptotic optimal competitive ratios of $3/2$ and 2 , respectively. The proof of the latter uses a detailed analysis involving pq-trees [10]. The competitive ratios are asymptotic in n .

2 The offline problem

In this section, we examine the Network Construction with Ordered Constraints problem in the offline case. We are able to obtain the same lower bound as Angluin et al. [7] in the general connectivity constraints case.

Theorem 1. *If $P \neq NP$, the approximation ratio of the **Network Construction with Ordered Constraints** problem is $\Omega(\log n)$.*

Proof. We prove the theorem by reducing from the Hitting Set problem. Let (U, \mathcal{S}) be a hitting set instance, where $U = \{u_1, \dots, u_n\}$ is the universe, and

$\mathcal{S} = \{S_1, \dots, S_m\}$ is a set of subsets of U . A subset $H \subset U$ is called a hitting set if $H \cap S_i \neq \emptyset$. The objective of the Hitting Set problem is to minimize $|H|$. We know from [13,20] that the Hitting Set problem cannot be approximated by any polynomial time algorithm within a ratio of $o(\log n)$ unless $P=NP$. Here we show that the Network Construction problem is inapproximable better than an $O(\log n)$ factor by first showing that we can construct a corresponding Network Construction instance to any given Hitting Set instance, and then showing that if there is a polynomial time algorithm that can achieve an approximation ratio $o(\log n)$ to the Network Construction problem, then the Hitting Set problem can also be approximated within in a ratio of $o(\log n)$, which is a contradiction.

We first define a Network Construction instance, corresponding to a given Hitting Set instance (U, \mathcal{S}) , with vertex set $U \cup W$, where $W = \{w_1, \dots, w_{n^c}\}$ for some $c > 2$. Note that we use the elements of the universe of hitting set instance as a part of the vertex set of Network Construction instance. The ordered constraints are the union of the following two sets:

- $\{(u_i, u_j)\}_{1 \leq i < j \leq n}$;
- $\{(S_k, w_l)\}_{S_k \in \mathcal{S}, 1 \leq l \leq n^c}$,

where by (S_k, w_l) we mean an ordered constraint with all vertices except the last one from a subset S_k of U , while the last vertex w_l is an element in W . The vertices from S_k are ordered arbitrarily.

We note that the first set of ordered constraints forces a complete graph on U , and the second set of ordering demands that there is at least one edge going out from each S_k connecting each element in W . More specifically let E_l denote the set of edges incident to w_l belonging to any solution to the Network Construction instance. Because of the second set of ordered constraints, the set $H_l = \{u \in U \mid \{u, w_l\} \in E_l\}$ is a hitting set of \mathcal{S} !

Let $H \subset U$ be any optimal solution to the hitting set instance, and denote by OPT_H the size of H , it is easy to see the two sets of ordered constraints can be satisfied by putting a complete graph on U and a complete bipartite graph between H and W . Hence the optimal solution to the Network Construction instance satisfies

$$\text{OPT} \leq \binom{n}{2} + n^c \text{OPT}_H,$$

where OPT is the minimum number of edges needed to solve the Network Construction instance. Let us assume that there is a polynomial time approximation algorithm to the Network Construction problem that adds ALG edges. Without loss of generality we can assume that the algorithm add no edge among vertices in W , because any edge within W can be removed without affecting the correctness of the solution, which implies that $\text{ALG} = \binom{n}{2} + \sum_{l=1}^{n^c} |E_l|$. Now if ALG is in the order $o(\log n \text{OPT})$, from the fact that $|H_l| = |E_l|$, we get

$$\begin{aligned} \min_{1 \leq l \leq n^c} |H_l| &\leq \frac{\text{ALG} - \binom{n}{2}}{n^c} = \frac{o(\log n (\binom{n}{2} + n^c \text{OPT}_H)) - \binom{n}{2}}{n^c} \\ &= o(\log n \text{OPT}_H), \end{aligned}$$

which means by finding the smallest set H_{l_0} among all the H_l s, we get a hitting set that has size within an $o(\log n)$ factor of the optimal solution to the Hitting Set instance, which is a contradiction. \square

We also observe that the upper bound from the more general problem implies a bound in our ordered case. We note the upper and lower bounds match when $r = \text{poly}(n)$.

Corollary 1 (of Theorem 2 from Angluin et al. [7]). *There is a polynomial time $O(\log r + \log n)$ -approximation algorithm for the Network Construction with Ordered Constraints problem on n nodes and r constraints.*

Proof. Observing that r ordered constraints imply at most nr unordered constraints on a graph with n nodes, we can use the $O(\log r)$ upper bound from Angluin et al. [7]. \square

3 The online problem

Here, we study the online problem, where constraints come in one at a time, and the algorithm must satisfy them by adding edges as the constraints arrive.

3.1 Arbitrary graphs

Theorem 2. *The competitive ratio for **Online Network Construction with Ordered Constraints** problem on n nodes and r ordered constraints has an upper bound of $O((\log r + \log n) \log n)$ against an oblivious adversary.*

Proof. To prove the statement, we first define the **Fractional Network Construction** problem, which has been shown by Angluin et al. [7] to have an $O(\log n)$ -approximation algorithm. The upper bound is then obtained by applying a probabilistic rounding scheme to the fractional solution given by the approximation. The proof heavily relies on arguments developed by Buchbinder and Noar [11], and Angluin et al. [7].

In the **Fractional Network Construction** problem, we are also given a set of vertices and a set of constraints $\{S_1, \dots, S_r\}$ where each S_i is a subset of the vertex set. Our task is to assign weights w_e to each edge e so that the maximum flow between each pair of vertices in S_i is at least 1. The optimization problem is to minimize $\sum w_e$. Since subgraph connectivity constraint is equivalent to requiring a maximum flow of 1 between each pair of vertices with edge weight $w_e \in \{0, 1\}$, the fractional network construction problem is the linear relaxation of the subgraph connectivity problem. Lemma 2 of Angluin et al. [7] gives an algorithm that multiplicatively updates the edge weights until all the flow constraints are satisfied. It also shows that the sum of weights given by the algorithm is upper bounded by $O(\log n)$ times the optimum.

As we pointed out in the introduction, an ordered constraint \mathcal{O} is equivalent to a sequence of subgraph connectivity constraints. So in the first step, we feed

the r sequences of connectivity constraints, each one is equivalent to an ordered constraint, to the approximation algorithm to the fractional network construction problem and get the edge weights. Then we apply a rounding scheme similar to the one considered by Buchbinder and Noar [11] to the weights. For each edge e , we choose t random variables $X(e, i)$ independently and uniformly from $[0, 1]$, and let the threshold $T(e) = \min_{i=1}^t X(e, i)$. We add e to the graph if $w_e \geq T(e)$.

Since the rounding scheme has no guarantee to produce a feasible solution, the first thing we need to do is to determine how large t should be to make all the ordered constraints satisfied with high probability.

We note that an ordered constraint $\mathcal{O}_i = \{v_{i1}, v_{i2}, \dots, v_{is_i}\}$ is satisfied if and only if the $(s - 1)$ connectivity constraints $\{v_{i1}, v_{i2}\}, \dots, \{v_{i1}, \dots, v_{is_i-1}, v_{is_i}\}$ are satisfied which is equivalent, in turn, to the fact that there is an edge that goes across the $(\{v_{i1}, \dots, v_{ij-1}\}, \{v_{ij}\})$ cut, for $2 \leq j \leq s_i$. For any fixed cut C , the probability the cut is not crossed equals $\prod_{e \in C} (1 - w_e)^t \leq \exp(-t \sum_{e \in C} w_e)$. By the max-flow min-cut correspondence, we know that $\sum_{e \in C} w_e \geq 1$ in the fractional solution given by the approximation algorithm for all cut $C = (\{v_{i1}, \dots, v_{ij-1}\}, \{v_{ij}\})$, $1 \leq i \leq r$, $2 \leq j \leq s_i$, and hence the probability that there exists at least one unsatisfied \mathcal{O}_i is upper bounded by $rn \exp(-t)$. So $t = c(\log n + \log r)$, for any $c > 1$, makes the probability that the rounding scheme fails to produce a feasible solution approaches 0 as n increases.

Because the probability that e is added equals the probability that at least one $X(e, i)$ is less than w_e , and hence is upper bounded by $w_e t$, we get the expected number of edges added is upper bounded by $t \sum w_e$ by linearity of expectation. Since the fractional solution is upper bounded by $O(\log n)$ times the optimum of the fractional problem, which is upper bounded by any integral solution, our rounding scheme gives a solution that is $O((\log r + \log n) \log n)$ times the optimum. \square

Corollary 2. *If the number of ordered constraints $r = \text{poly}(n)$, then the algorithm above gives a $O((\log n)^2)$ upper bound for the competitive ratio against an oblivious adversary.*

Remark 1. We can generalise theorem 2 to the weighted version of the Online Network Construction with Ordered Constraints problem. In the weighted version, each edge $e = (u, v)$ is associated with a cost c_e and the task is to select edges such that the connectivity constraints are satisfied and $\sum c_e w_e$ is minimised where $w_e \in \{0, 1\}$ is a variable indicating whether an edge is picked or not and c_e is the cost of the edge. The same approach in the proof of Theorem 2 gives an upper bound of $O((\log r + \log n) \log n)$ for the competitive ratio of the weighted version of the Online Network Construction with Ordered Constraints problem.

Theorem 3. *This is a $\Omega(\log n)$ lower bound for the competitive ratio for the **Online Network Construction with Ordered Constraints** problem against an oblivious adversary.*

Proof. The adversary divides the vertex set into two parts U and V , where $|U| = \sqrt{n}$ and $|V| = n - \sqrt{n}$, and gives the constraints as follows. Firstly, it

forces a complete graph in U by giving the constraint $\{u_i, u_j\}$ for each pair of vertices $u_i, u_j \in U$. At this stage both the algorithm and optimal solution will have a clique in U , which costs $\Theta(n)$.

Then, for each $v \in V$, first fix a random permutation π_v on U and give the ordered constraint

$$\mathcal{O}_{(v,i)} = (\pi_v(1), \pi_v(2), \dots, \pi_v(i), v).$$

First note that all these constraints can be satisfied by adding $e_v = \{\pi_v(1), v\}$ for each $v \in V$ which costs $\Theta(n)$. However, the adversary gives constraints in the following order:

$$\mathcal{O}_{(v,\sqrt{n})}, \mathcal{O}_{(v,\sqrt{n}-1)}, \dots, \mathcal{O}_{(v,1)}.$$

We now claim that for each $v \in V$, the algorithm will add $\Omega(\log n)$ edges in expectation. This is because each edge added by the algorithm is a random guess for $\pi_v(1)$ and this edge cuts down the number of unsatisfied $\mathcal{O}_{(v,i)}$ by half in expectation. This means the algorithm adds $\Omega(n + n \log n)$ edges. This gives us the desired result because $\text{OPT} = O(n)$. \square

Now we study the online problem when it is known that an optimal graph can be a star or a path. These special cases are challenging in their own right and are often studied in the literature to develop more general techniques [7].

3.2 Stars

Theorem 4. *The optimal competitive ratio for the **Online Network Construction with Ordered Constraints** problem when the algorithm knows that an optimal solution forms a **star** is asymptotically $3/2$.*

Proof. For lower bound, we note that the adversary can simply give $\mathcal{O}_i = (v_1, v_2, v_i)$, $3 \leq i \leq n$ obviously for the first $n - 2$ rounds. Then an algorithm, besides adding $\{v_1, v_2\}$ in the first round, can only choose from adding either $\{v_1, v_i\}$ or $\{v_2, v_i\}$, or both in each round. After the first $n - 2$ rounds, the adversary counts the number of v_1 and v_2 's neighbors, and chooses the one with fewer neighbors, say v_1 , to be the center by adding (v_1, v_i) for some $3 \leq i \leq n$. Since the algorithm has to add at least $\lceil (n - 2)/2 \rceil$ edges that are unnecessary in the hindsight, we get an asymptotic lower bound $3/2$.

For upper bound, assume that the first ordered constraint is \mathcal{O}_1 is (v_1, v_2, \dots) , the algorithm works as follows:

1. It adds $\{v_1, v_2\}$ in the first round.
2. Then for any constraint that starts with v_1 and v_2 , it splits the remaining vertices in the constraint (other than v_1 and v_2) into two sets of sizes differing by at most 1, and connects each vertex in first set to v_1 and each vertex in the other set to v_2 .
3. Upon seeing a constraint that does not start with v_1 and v_2 , which reveals the center of the star, it connects the center to all vertices that are not yet connected to the center.

Since the algorithm adds, at most $n/2 - 1$ edges to the wrong center, this gives us an asymptotic upper bound $3/2$, which matches the lower bound. \square

3.3 Paths

In the next two theorems, we give matching lower and upper bounds (in the limit) for path graphs.

Theorem 5. *The competitive ratio for the **Online Network Construction with Ordered Constraints** problem when the algorithm knows that the optimal solution forms a **path** has an asymptotic lower bound of 2.*

Proof. Fix an arbitrary ordering of the vertices $\{v_1, v_2, v_3, \dots, v_n\}$. For $3 \leq i \leq n$, define the **pre-degree** of a vertex v_i to be the number of neighbors v_i has in $\{v_1, v_2, v_3, \dots, v_{i-1}\}$. Algorithm 1 below is a simple strategy the adversary can take to force v_3, \dots, v_n to all have pre-degree at least 2. Since any algorithm will add at least $2n - 3$ edges, this gives an asymptotic lower bound of 2. Suppose

Algorithm 1 Forcing pre-degree to be at least 2

```

Give ordered constraint  $\mathcal{O} = (v_1, v_2, v_3, \dots, v_n)$  to the algorithm;
for  $i = 3$  to  $n$  do
    if the pre-degree of  $v_i$  is at least 2 then
        continue;
    else
        pick up at random a path (say  $P_i$ ) that satisfies all the constraints up to this
        round and an endpoint  $u$  of the path that is not connected to  $v_i$ , and gives the
        algorithm the constraint  $(v_i, u)$ ;
    end if
end for

```

P_i was the path picked in round i (i.e. P_i satisfies all constraints upto round i). Then, P_i along with the edge (v_i, u) is a path that satisfies all constraints upto round $i + 1$. Hence by induction, for all i , there is a path that satisfies all constraints given by the adversary upto round i . \square

Theorem 6. *The competitive ratio for the **Online Network Construction with Ordered Constraints** problem when the algorithm knows that the optimal solution forms a **path** has an asymptotic upper bound of 2.*

Proof. For our algorithm matching the upper bound, we use the pq-trees, introduced by Booth and Lueker [10], which keep track all consistent permutations of vertices given contiguous intervals of vertices. Our analysis is based on ideas from Angluin et al. [7], who also use pq-trees for analyzing the general problem.¹

A **pq-tree** is a tree whose leaf nodes are the vertices and each internal node is either a **p-node** or a **q-node**.

- A **p-node** has a two or more children of any type. The children of a p-node form a contiguous interval that can be in any order.

¹ Angluin et al. [7] have a small error in their argument because their potential function fails to explicitly consider the number of p-nodes, which creates a problem for some of the pq-tree updates. We fix this, without affecting their asymptotic bound. For the ordered constraints case, we are also able to obtain a much finer analysis.

- A **q-node** has three or more children of any type. The children of a q-node form a contiguous interval, but can only be in the given order of its inverse.

Every time a new interval constraint comes, the tree update itself by identifying any of the eleven patterns, P0, P1, ..., P6, and Q0, Q1, Q2, Q3, of the arrangement of nodes and replacing it with each correspondent replacement. The update fails when it cannot identify any of the patterns, in which case the contiguous intervals fail to produce any consistent permutation. We refer readers to Section 2 of Booth and Lueker [10] for a more detailed description of pq-trees.

The reason we can use a pq-tree to guide our algorithm is because of an observation made in Section 1 that each ordered constraint $(v_1, v_2, v_3, \dots, v_{k-1}, v_k)$ is equivalent to k interval constraints $\{v_1, v_2\}, \{v_1, v_2, v_3\}, \dots, \{v_1, \dots, v_{k-1}\}, \{v_1, \dots, v_{k-1}, v_k\}$. So upon seeing one ordered constraints, we reduce the pq-tree with the equivalent interval constraints, *in order*. Then what our algorithm does is simply to add edge(s) to the graph every time a pattern is identified and replaced with its replacement, so that the graph satisfies all the seen constraints. Note that to reduce the pq-tree with one interval constraint, there may be multiple patterns identified and hence multiple edges may be added.

Before running into details of how the patterns determine which edge(s) to add, we note that, without loss of generality, we can assume that the the algorithm is in either one of the following two stages.

- The pq-tree is about to be reduced with $\{v_1, v_2\}$.
- The pq-tree is about to be reduced with $\{v_1, \dots, v_k\}$, when the reductions with $\{v_1, v_2\}, \dots, \{v_1, \dots, v_{k-1}\}$ have been done.

Because of the structure of constraints discussed above, we do not encounter all pq-tree patterns in their full generality, but in the special forms demonstrated in Table 1. Based on this, we make three important observations which can be verified by carefully examining how a pq-tree evolves along with our algorithm.

1. The only p-node that can have more than two children is the root.
 2. At least one of the two children of a non-root p-node is a leaf node.
 3. For all q-nodes, there must at least one leaf node in any two adjacent children.
- Hence, Q3 doesn't appear.

Now we describe how the edges are going to be added. Note that a pq-tree inherently learns edges that appear in optimum even when those edges are not forced by constraints. Apart from adding edges that are necessary to satisfy the constraints, our algorithm will also add any edge that the pq-tree inherently learns. For all the patterns except Q2 such that a leaf node v_k is about to be added as a child to a different node, we can add one edge joining v_k to v_{k-1} . For all such patterns except Q2, it is obvious that this would satisfy the current constraint and all inherently learnt edges are also added. For Q2, the pq-tree could learn two edges. The first edge is (v_k, v_{k-1}) . The second one is an edge between the leftmost subtree of the daughter q-node (call T_l) and the node to its left (call v_l). Based on Observation 3, v_l is a leaf. But based on the algorithm, one of these two edges is already added. Hence, we only need to add one edge when Q2 is applied. For P5, we add the edge as shown in Table 1.

	Pattern	Replacement
P2		
P3		
P4(1)		
P4(2)		
P5		
P6(1)		
P6(2)		
Q2		

Table 1: Specific patterns and replacements that appear through the algorithm. P4(1) denotes the case of P4 where the top p-node is retained in the replacement and P4(2) denotes the case where the top p-node is deleted. The same is true for P6. P0, P1, Q0, and Q1 are just relabelling rules, and we have omitted them because no edges need to be added. We use the same shapes to represent p-nodes, q-nodes, and subtrees as in Booth and Lueker’s paper [10] for easy reference, and we use diamonds to represent leaf nodes.

Let us denote by P and Q the sets of p-nodes and q-nodes, respectively, and by $c(p)$ the number of children node p has. And let potential function ϕ of a tree T be defined as

$$\phi(T) = a \sum_{p \in P} c(p) + b|P| + c|Q|,$$

where a , b , and c are coefficients to be determined later.

	$\sum_{p \in P} c(p)$	$ P $	$ Q $	$-\Delta\Phi$	number of edges added
P2	1	1	0	$-a - b$	1
P3	-2	-1	1	$2a + b - c$	0
P4(1)	-1	0	0	a	1
P4(2)	-2	-1	0	$2a + b$	1
P5	-2	-1	0	$2a + b$	1
P6(1)	-1	0	-1	$a + c$	1
P6(2)	-2	-1	-1	$2a + b + c$	1
Q2	0	0	-1	c	1
Q3	0	0	-2	$2c$	1

Table 2: How the terms in the potential function: $\sum_{p \in P} c(p)$, $|P|$, and $|Q|$ change according to the updates.

We want to upper bound the number of edges added for each pattern by the drop of potential function. We collect the change in the three terms in the potential function that each replacement causes in Table 2, and we can solve a simple linear system to get that choosing $a = 2$, $b = -3$, and $c = 1$ is sufficient. For ease of analysis, we add a dummy vertex v_{n+1} that does not appear in any constraint. Now, the potential function starts at $2n - 1$ (a single p-node with $n + 1$ children) and decreases to 2 when a path is uniquely determined. Hence, the number of edges added by the algorithm is $2n - 3$, which gives the desired asymptotic upper bound. \square

References

1. Noga Alon and Vera Asodi. Learning a hidden subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.
2. Noga Alon, Baruch Awerbuch, and Yossi Azar. The online set cover problem. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 100–105. ACM, 2003.
3. Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Seffi Naor. A general approach to online network optimization problems. *ACM Transactions on Algorithms (TALG)*, 2(4):640–660, 2006.
4. Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004.
5. Dana Angluin, James Aspnes, and Lev Reyzin. Optimally learning social networks with activations and suppressions. In *International Conference on Algorithmic Learning Theory*, pages 272–286. Springer, 2008.

6. Dana Angluin, James Aspnes, and Lev Reyzin. Inferring social networks from outbreaks. In *International Conference on Algorithmic Learning Theory*, pages 104–118. Springer, 2010.
7. Dana Angluin, James Aspnes, and Lev Reyzin. Network construction with sub-graph connectivity constraints. *Journal of Combinatorial Optimization*, 29(2):418–432, 2015.
8. Dana Angluin and Jiang Chen. Learning a hidden graph using $o(\log n)$ queries per edge. *Journal of Computer and System Sciences*, 74(4):546–556, 2008.
9. Richard Beigel, Noga Alon, Simon Kasif, Mehmet Serkan Apaydin, and Lance Fortnow. An optimal procedure for gap closing in whole genome shotgun sequencing. In *Proceedings of the fifth annual international conference on Computational biology*, pages 22–30. ACM, 2001.
10. Kellogg S Booth and George S Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of Computer and System Sciences*, 13(3):335–379, 1976.
11. Niv Buchbinder and Joseph Naor. The design of competitive online algorithms via a primal: dual approach. *Foundations and Trends® in Theoretical Computer Science*, 3(2–3):93–263, 2009.
12. Gregory Chockler, Roie Melamed, Yoav Tock, and Roman Vitenberg. Constructing scalable overlays for pub-sub with many topics. In *Proceedings of the twenty-sixth annual ACM symposium on Principles of distributed computing*, pages 109–118. ACM, 2007.
13. Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.
14. Manuel Gomez-Rodriguez, Jure Leskovec, and Andreas Krause. Inferring networks of diffusion and influence. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 5(4):21, 2012.
15. Vladimir Grebinski and Gregory Kucherov. Reconstructing a hamiltonian cycle by querying the graph: Application to dna physical mapping. *Discrete Applied Mathematics*, 88(1):147–165, 1998.
16. Anupam Gupta, Ravishankar Krishnaswamy, and R Ravi. Online and stochastic survivable network design. *SIAM Journal on Computing*, 41(6):1649–1672, 2012.
17. Ephraim Korach and Michal Stern. The clustering matroid and the optimal clustering tree. *Mathematical Programming*, 98(1-3):385–414, 2003.
18. Ephraim Korach and Michal Stern. The complete optimal stars-clustering-tree problem. *Discrete Applied Mathematics*, 156(4):444–450, 2008.
19. Hervé Moulin and Francois Laigret. Equal-need sharing of a network under connectivity constraints. *Games and Economic Behavior*, 72(1):314–320, 2011.
20. Ran Raz and Shmuel Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability pcg characterization of np. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 475–484. ACM, 1997.
21. Lev Reyzin and Nikhil Srivastava. Learning and verifying graphs using queries with a focus on edge counting. In *International Conference on Algorithmic Learning Theory*, pages 285–297. Springer, 2007.
22. Kazumi Saito, Ryohei Nakano, and Masahiro Kimura. Prediction of information diffusion probabilities for independent cascade model. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 67–75. Springer, 2008.